

Situational Object Boundary Detection

J.R.R. Uijlings
University of Edinburgh

V. Ferrari
University of Edinburgh

Abstract

Intuitively, the appearance of true object boundaries varies from image to image. Hence the usual monolithic approach of training a single boundary predictor and applying it to all images regardless of their content is bound to be suboptimal. In this paper we therefore propose situational object boundary detection: We first define a variety of situations and train a specialized object boundary detector for each of them using [10]. Then given a test image, we classify it into these situations using its context, which we model by global image appearance. We apply the corresponding situational object boundary detectors, and fuse them based on the classification probabilities. In experiments on ImageNet [35], Microsoft COCO [24], and Pascal VOC 2012 segmentation [13] we show that our situational object boundary detection gives significant improvements over a monolithic approach. Additionally, our method substantially outperforms [17] on semantic contour detection on their SBD dataset.

1. Introduction

Most methods for object boundary detection are monolithic and use a single predictor to predict all object boundaries in an image [2, 10, 23] regardless of the image content. But intuitively, the appearance of object boundaries is dependent on what is depicted in the image. For example, black-white transitions are often good indicators of object boundaries, unless the image depicts a zebra as in Figure 1. Outdoors, the sun may cast shadows which create strong contrasts that are not object boundaries, while similar colour contrasts in an indoor environment with diffuse lighting may be caused by object boundaries. Furthermore, not all objects are equally important in all circumstances: one may want to detect the boundary between a snowy mountain and the sky in images of winter holidays, while ignoring sky-cloud transitions in images depicting air balloons, even though such boundaries may be visually very similar. These examples show that one cannot expect a monolithic predictor to accurately predict object boundaries in all situations.

In this work we recognize the need for different object boundary detectors in different situations: first we define a

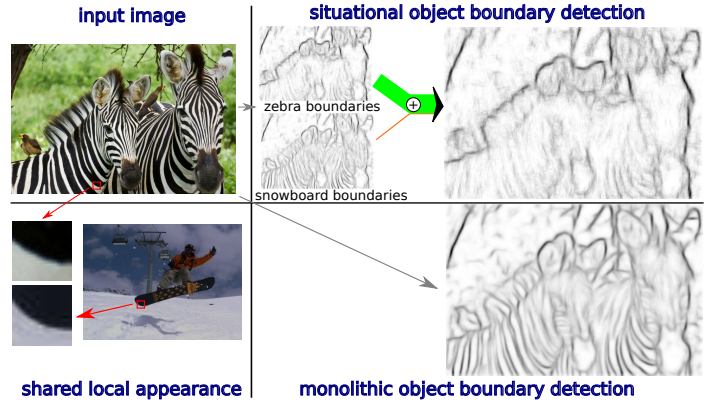


Figure 1. Monolithic vs situational object boundary detection. Black-white transitions indicate an object boundary for the snowboard, but are false object boundaries for a zebra. This ambiguity cannot be resolved by a monolithic detector. In contrast, by training class specific object boundary detectors and classifying the image as a zebra, we correctly ignore most of the stripes.

set of situations and pre-train object boundary detectors for each of them. For a test image, we classify which situations the image depicts based on its context, modelled by global image appearance. Then we apply the appropriate set of object boundary detectors. Hence conditioned on the situation of an image we choose which object boundary detectors to run. We call this Situational Object Boundary Detection.

One important question is how to define such situations. Since the appearance of object boundaries are for a large part dependent on the object class, one natural choice is to use each object class as a single situation. This results in *class specific* object boundary detectors, which can deal for example with the zebra in Figure 1. However, object boundaries are also determined by the object pose and the background or context of the image. Since this can vary within a single object class, we propose to cluster images of a single class into subclasses based on global image appearance. This leads to *subclass specific* object boundary detectors. Finally, one can imagine that the context of the image itself determines what kind of object boundaries to expect. For example, one can expect cow-grass boundaries in the countryside and street-car boundaries in the city. Therefore we cluster images based on their global image appearance, which results in *class agnostic* object boundary detectors.

Hence we experiment with three types of situations: *class specific*, *subclass specific*, and *class agnostic*.

Obviously, situational object boundary detection requires more training data than a monolithic approach. Therefore we cannot use the standard BSD500 [2] dataset of 500 images for our evaluation. Instead, we evaluate on three larger datasets: Pascal VOC 2012 segmentation [13], Microsoft COCO [24], and part of ImageNet [35]. Microsoft COCO is two orders of magnitude larger than BSD500. For ImageNet we train from segments which are created in a semi-supervised fashion by Guillaumin et al. [16].

Additionally, our class-specific situational object boundary detectors can also be applied to semantic contour detection, the task of predicting class-specific object boundaries [17]. We compare with [17] on their SBD dataset.

2. Related Work

Manually defined predictors. Early work on object boundary detection aimed to manually define local filters to generate edges from an image. In these works, convolutional derivative filters are applied to find local image gradients [12, 32, 34] and their local maximum [6, 28].

Trained predictors. But object boundaries arise from a complex combination of local cues. Therefore more recent techniques resort to machine learning and datasets with annotated object boundaries: Martin et al. [29] compute local brightness, colour, and texture cues, which they combine using a logistic model. Both Mairal et al. [27] and Prasad et al. [31] use RGB-features from local patches centred on edges found by the canny edge detector [6], which they classify as true or false positives. Dollár et al. [9] use boosted decision trees to predict if the centre label of an image patch is an object boundary or not. Lim et al. [23] use Random Forests [5] to predict sketch tokens, which are object boundary *patches* generated by k-means clustering. Dollár and Zitnick [10] proposed structured random forests, which use object boundary patches as structured output labels inside a random forest. Their method is extremely fast and yields state-of-the-art results. We build on [10] in our paper.

Domain specific predictors. Some works that use machine learning to predict object boundaries observed that this enables tuning detectors to specific domains. Dollár et al. [9] showed qualitative examples of domain-specific detectors for finding mouse boundaries in a laboratory setting and detecting streets in aerial images. Both [27] and [31] used class-specific object boundary detectors for boundary-based object classification. Whereas in all these cases the domain was predefined, in this work we automatically choose which object boundary detector to apply at runtime.

Semantic contour detection. Like [27] and [31], Hariharan et al. [17] addressed class-specific object boundary detection. They call this ‘semantic contour detection’ and cre-

ate the SBD benchmark to directly evaluate this task. Their method combines a monolithic object boundary detector (gPb [2]) with object class detectors (Poselets [4]). Since the class-specific version of our situational object boundary detection can readily be applied to semantic contour detection, we compare to [17] in Section 4.4.

Globally constrained predictors. Instead of predicting boundaries only at a local level, Arbeláez et al. [2] cast the problem into a global optimization framework capturing non-local properties in the spirit of Normalized Cuts [36]. In this paper we use the global image appearance to determine the set of local object boundary predictors to use. In this sense, the global appearance of the image restricts our algorithm to a limited set of expected object boundaries.

Contextual guidance. Context, as modelled by global image appearance, has been successfully used to guide a variety of computer vision tasks. Torralba et al. [38] showed that global image features effectively constrain both the object class and its location, which is frequently used in object localisation (e.g. [13, 14, 18]). Boix et al. [3] do semantic segmentation by region prediction, where the global image appearance enforces a consistency potential in their hierarchical CRF. Liu et al. [25] perform semantic segmentation through label transfer. Given a test image, they retrieve nearest neighbours from a pixel-wise annotated dataset using global image appearance. After region alignment, they transfer labels to the test image. In this paper we use context modelled by global image features to select those object boundary detectors that correspond to the situation depicted in the image.

3. Method

3.1. Situational Object Boundary Detection

Our main idea is visualized in Figure 2. For each specific situation, one can train a specialized object boundary detector. Given a test image, one then only needs to apply those boundary detectors which best fit its situation. Intuitively, the global image appearance can help distinguish the local appearance of true object boundaries from edges caused by other phenomena.

Formally, let $\mathcal{D} = \{D_1, \dots, D_k\}$ be a set of k trained object boundary detectors for a corresponding set of k situations $\mathcal{S} = \{S_1, \dots, S_k\}$. Applying the j -th detector D_j to image I gives the boundary prediction $D_j(I)$. We write the probability that image I corresponds to situation S_j as $P(S_j|I)$, which we obtain using global image classification as explained in section 3.3. Now we get the final object boundary prediction $\mathbf{D}(I)$ by:

$$\mathbf{D}(I) = \sum_{j=1}^k P(S_j|I) \cdot D_j(I) \quad (1)$$

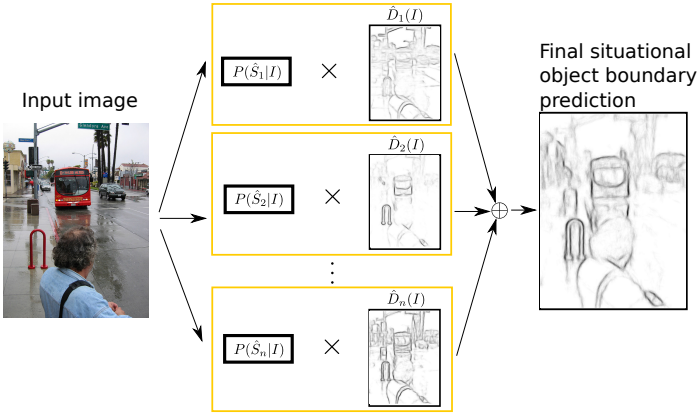


Figure 2. Overview of situational object boundary detection. For each situation there is a specialised boundary detector \hat{D}_j which we apply by $\hat{D}_j(I)$. The specialised predictions vary greatly and are combined into a final prediction using Equation (2).

Of course, we do not need to apply all object boundary detectors to the image since $P(S_j|I)$ is likely to be small for most situations j . To reduce computational costs we take the top few $n \ll k$ situations for which $P(S_j|I)$ is highest. Formally, let $\hat{\mathcal{S}} = \{\hat{S}_1, \dots, \hat{S}_n\}$ be an ordered set for a specific image I such that $P(\hat{S}_i|I) > P(\hat{S}_j|I)$ for all $i < j$. Let $\hat{\mathcal{D}}$ be the set of boundary detectors corresponding to $\hat{\mathcal{S}}$. Then the final object boundary prediction is obtained by:

$$\mathbf{D}(I) = \frac{1}{Z} \sum_{j=1}^n P(\hat{S}_j|I) \cdot \hat{D}_j(I) \quad (2)$$

where $Z = \sum_{j=1}^n P(\hat{S}_j|I)$ is a normalizing factor ensuring that the values of the predicted boundaries are comparable for different n and across images.

We have two choices for n : either we fix n or we take n such that $Z > m$ for a specific probability mass m . We determine the best solution experimentally in section 4.1.

3.2. Situations

For situational object boundary detection to work, the key is to define proper situations. We propose three ways to define our situations as visualised in Figure 3: *class specific*, *subclass specific*, and *class agnostic*.

Class specific. As the term already says itself, object boundaries are caused by the presence of an object. A logical way to define a situation is therefore to use class specific situations, leading to class specific boundary detection. We use class labels from the dataset to obtain these situations.

Class specific situations constrain the appearance of object boundaries in two ways. Most importantly, instances of the same class tend to have similar appearance: in Figure 3a the boundaries of a baboon are all a specific type of fur, while air balloons have a characteristic oval shape. Second, objects often occur in similar contexts: killer-whales are mostly in the water while balloons are often in the air. If both the context and object class is the same, there is little variation in the appearance of object boundaries and one

can learn an object boundary detector which is sensitive to these specific object boundaries.

Subclass specific. For some classes, its instances are depicted in a variety of contexts, poses, and from a variety of viewpoints, which can significantly influence the appearance of the object boundaries. Take for example the killer-whale in Figure 3b. Photographed in the wild the object boundaries are only caused by water-whale transitions, while in a whale-show object boundaries can also be caused by crowd-whale transitions. Furthermore, spurious edges caused by the crowd should not yield object boundaries here. Additionally, a viewpoint from within the water or from above the water causes the object boundaries to be very different due to colour changes and absence/presence of foaming water or waves. Pose may also affect object boundary appearance: a sleeping, curled-up cat has much smoother boundaries than a playing cat.

We create subclass specific situations by taking all images of a certain class, model their global image appearance as described in Section 3.3, and apply k-means clustering.

Class agnostic. Finally, the appearance of object boundaries may be more influenced by context than by the object class itself. For example, as visualised in Figure 3c, photographs taken through a fence yield spurious edges which are not object boundaries. Detecting such situation allows for using an object boundary detector which ignores edges from this fence. Furthermore, various object classes occur in similar contexts and share characteristics. Indeed, the second row shows furry animals in a forest environment, giving rise to a similar appearance of object boundaries.

Therefore the last situation type we consider is class agnostic. We ignore all class labels and cluster all images of the training set using k-means on global image appearance. As shown in Figure 3c, this leads to clusters of objects in similar contexts, some with predominantly instances of a single class.

3.3. Image Classification

For each situation $S_j \in \mathcal{S}$ we need to predict $P(S_j|I)$. We do this using either Bag of Visual Words [8, 37] or Convolutional Neural Net (CNN) features [21].

Bag of Visual Words. We extract SIFT descriptors [26] of 16×16 pixels on a dense regular grid [20] at every 4 pixels using [39]. We use PCA to reduce SIFT to 84 dimensions. We train a GMM with diagonal covariance of 64 clusters. We then create Fisher Vectors following [30]: we use derivatives only with respect to the means and standard deviations of the GMM. Vectors are normalized by taking the square root while keeping the sign, followed by L2 norm. We use a spatial partitioning [22] using the whole image and a division into three horizontal regions (e.g. [39]). The final Fisher representation has 43008 dimensions.

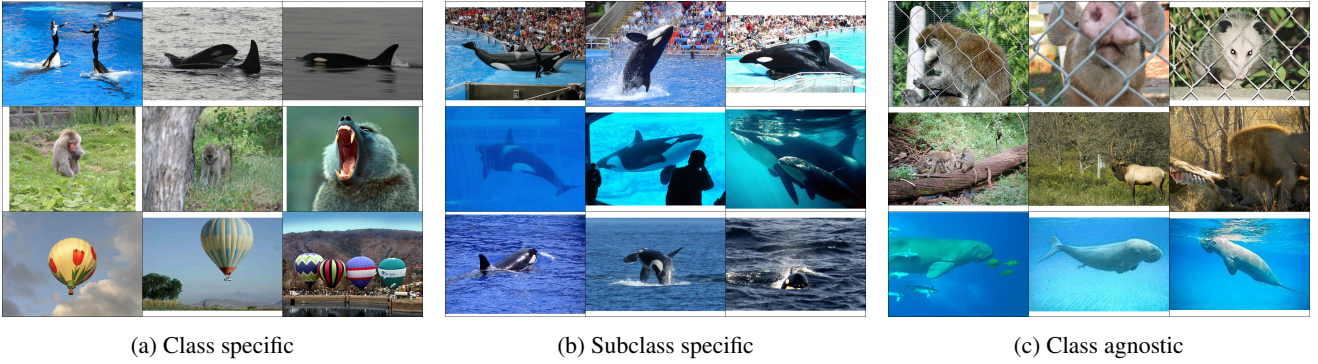


Figure 3. Visualisation for the three types of situations used in this paper. Each row per subfigure depicts three example images of a single situation on ImageNet (Section 4.1). Figure 3a shows class specific situations, where each situation is a single object class. Figure 3b show subclass specific situations, beneficial for classes with significant context or pose variation such as the killer-whale. Finally, Figure 3c shows class agnostic situations, which results in contextually similar clusters, some containing predominantly images of a single class.

CNN features. We use the publicly available software for deep Neural Networks of Jia et al. [19]. Instead of training a specialized network for each dataset, we choose the more flexible option of using a pre-trained network, removing the final classification layer, and using the last layer as global image appearance features. This was shown to yield excellent features by e.g. [11, 15, 33].

In particular, we use the pre-trained network modelled after Krizhevsky [21] that comes with [19], trained on the training set of the ILSVRC classification task [35]. This network takes as input RGB images rescaled to 227×227 pixels. It consists of five convolutional layers, two fully connected layers, and a final classification layer which we discard. Hence we use the outputs of the 7-th layer as CNN features, yielding features of 4096 dimensions.

Classification. For both the Fisher Vectors and CNN features, we train linear SVMs with Stochastic Gradient Descent using [40]. We use cross-validation to optimize the slack-parameter and, following [1], to optimize the relative sampling frequency of positive examples.

3.4. Boundary Detector

As boundary detector we use the Structured Edge Forests of Dollár and Zitnick [10], as these are extremely fast and yield state-of-the-art performance. Using their standard settings, their detector predicts 16×16 pixel boundary masks from 32×32 pixel local image patches. From each local image patch a variety of colour and gradient features is extracted. They train a random forest directly on the structured output space of segmentation masks: at each node they sample 256 random pixel pairs and perform binary tests checking if both pixels come from the same segment. The resulting 256 dimensional vector is reduced to a single dimension using PCA, where its sign is used as a binary label. This allows for the calculation of information gain as usual.

Unless mentioned otherwise, we use their framework with standard settings except for the number of training patches. We lower these from 1 million to 300,000 resulting in similar performance as shown in Section 4.1.

4. Results

In Section 4.1 to 4.3, we evaluate our method on object boundary detection on ImageNet [35], Microsoft COCO [24], and Pascal VOC 2012 segmentation [13]. We use the evaluation software of [29], average results over all images and report precision/recall curves, precision at 20% and 50% recall, and average precision (AP).

In Section 4.4, we evaluate our method on semantic contour detection on the SBD database [17] using their evaluation software and report average precision (AP).

4.1. ImageNet

Dataset. While ImageNet has no manually annotated object boundaries, Guillaumin et al. [16] obtained good segmentations using a semi-supervised segmentation transfer strategy, applied to increasingly difficult image subsets. As our training set, we use their most reliable segmentations created from bounding box annotations. As test set, we use the ground-truth segmentations collected by [16].

To keep evaluation time reasonable we randomly sample 100 classes from the set of [16]. This results in 23,457 training and 1,000 test images. Since each image is annotated with one object class, this experiment evaluates only boundaries of that class.

Number of situations. For subclass specific situations, we choose to cluster classes into 10 subclasses, yielding 1000 situations. For good comparison, we choose to also have the same number of 1000 class agnostic situations.

Number of detectors at test time. We now establish the number of object boundary detectors to apply to get optimal performance using Equation (2). Table 1 shows results when varying n for subclass specific object boundary detection (other situations yield similar results). As can be seen, starting from $n = 5$ results saturate for both methods. Looking at the probability mass Z , at $n = 5$ it is 61% for Fisher vectors and 71% for CNN features. However, Z greatly differs per image. Hence for stable and efficient computation time with optimal performance, we fix $n = 5$

	$n = 1$	$n = 3$	$n = 5$	$n = 25$
Z - CNN - subclass specific	47%	65%	71%	85%
Z - Fisher - subclass specific	29%	51%	61%	79%
AP - CNN - subclass specific	0.274	0.289	0.296	0.295
AP - Fisher - subclass specific	0.267	0.283	0.290	0.291
AP - Monolithic	0.258	0.259	0.260	0.260

Table 1. Influence of number of situational object boundary detectors applied at test time. Results saturate in average precision (AP) after applying 5 object boundary detectors.

	precision at 20% recall	precision at 50% recall	average precision
monolithic	0.382	0.282	0.260
CNN - class specific	0.435	0.311	0.289
CNN - subclass specific	0.451	0.317	0.296
CNN - class agnostic	0.446	0.315	0.295
Fisher - class specific	0.426	0.305	0.283
Fisher - subclass specific	0.442	0.312	0.290
Fisher - class agnostic	0.429	0.307	0.284
GT - class specific	0.433	0.311	0.290
monolithic - CNN enhanced	0.385	0.278	0.259

Table 2. Results on ImageNet show that situational object boundary detection significantly outperforms a monolithic strategy.

random forest detectors (of 8 trees) for all subsequent experiments.

Baseline. Our baseline (*monolithic*) is a single monolithic detector. However, for a fair comparison our baseline should be trained on the same number of training patches and use the same number of decision trees. This is equivalent to training multiple monolithic detectors [10]. As shown in Table 1, results are affected little by training more monolithic detectors, and stabilize at $n = 5$ at 0.260 AP.

We also trained a random forest with the recommended 1M training examples [10] instead of 300k. This yields 0.262 AP. Since this is not significantly different, for consistency of all experiments we choose as baseline $n = 5$ random forests trained on 300k examples per tree.

Situational Object Boundary Detection. Figure 4 and Table 2 show that situational object boundary detection significantly outperforms the monolithic approach. Using CNN features, at 20% recall, the precision for monolithic is 0.38, while it is respectively 0.44, 0.45, and 0.45 for class specific, subclass specific, and class agnostic situations.

Figure 4 shows that subclass specific situations slightly outperform class specific situations. This is because subdivision into subclasses by clustering yields more specialized object boundary detectors, which are especially helpful when the object class can occur in different contexts. Indeed, looking at performance increase of individual classes, the use of subclasses yields an increase in AP of 0.04, 0.08, and 0.14 for respectively *killer-whale*, *airship*, and *basketball*. The variety of contexts of the killer-whale can be seen in Figure 3b, *airships* occur on the ground and against the sky, while basketball images range from basketball close-ups, to indoor competition (see Figure 5), to outdoor play.

Note that the monolithic boundary detector is trained exclusively on the objects of interest. Hence if a local image patch causes a false boundary prediction, it is necessarily

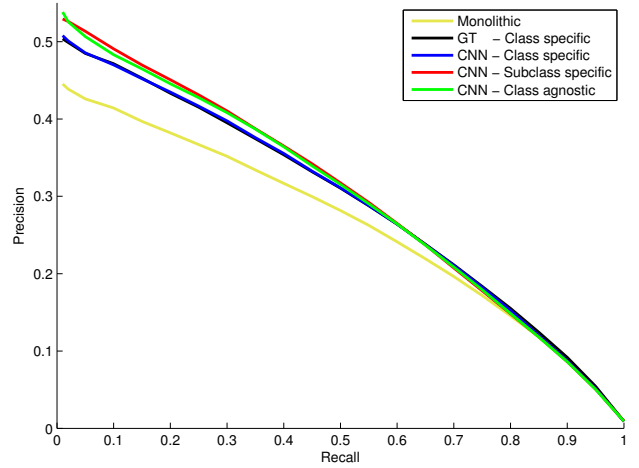


Figure 4. Performance of object boundary detection on ImageNet. Situational object boundary detection significantly outperforms monolithic. The black line is occluded by the blue.

similar in appearance to a local image patch of a true object boundary. Now notice in Figure 5 that monolithic boundary detection fires on many non-object boundary edges: the crowd of the basketball player, the shade behind the dog, the dog’s internal boundaries, and the water of the killer-whale. Therefore such background edges are necessarily similar in appearance to true object boundaries. This means a monolithic approach can never work well in all situations.

In contrast, situational object boundary detection performs much better, especially when using subclass specific situations. On the basketball image, our method ignores not only the crowd but also the player, which is good since the player is not the object of interest. For the dog our method focuses primarily on the dog boundaries ignoring shadow and its interior boundaries. For the killer-whale spurious edges caused by the water are ignored.

We conclude that by using object boundary detectors specialized for the identified situation, we effectively constrain the expected local appearance of object boundaries, which helps resolving ambiguities. This yields significant improvements: whereas a monolithic approach results in 0.260 AP, our subclass specific situation yield 0.296 AP, a relative improvement of 14%.

CNNs vs Fisher Vectors. Table 2 shows that CNN features work generally better than Fisher vectors for situational object boundary detection. This confirms other observations on the strength of CNN features (e.g. [7, 11, 33]). For class-agnostic situations improvements are especially good since it improves both the creation of situations and the classification. We use CNN features for the remainder of this paper.

Using ground-truth image labels. Table 2 includes an experiment where we use the ground-truth label to determine which class-specific boundary detector should be applied (*GT - class specific*). This helps assessing the quality of the global image appearance classifier within our frame-

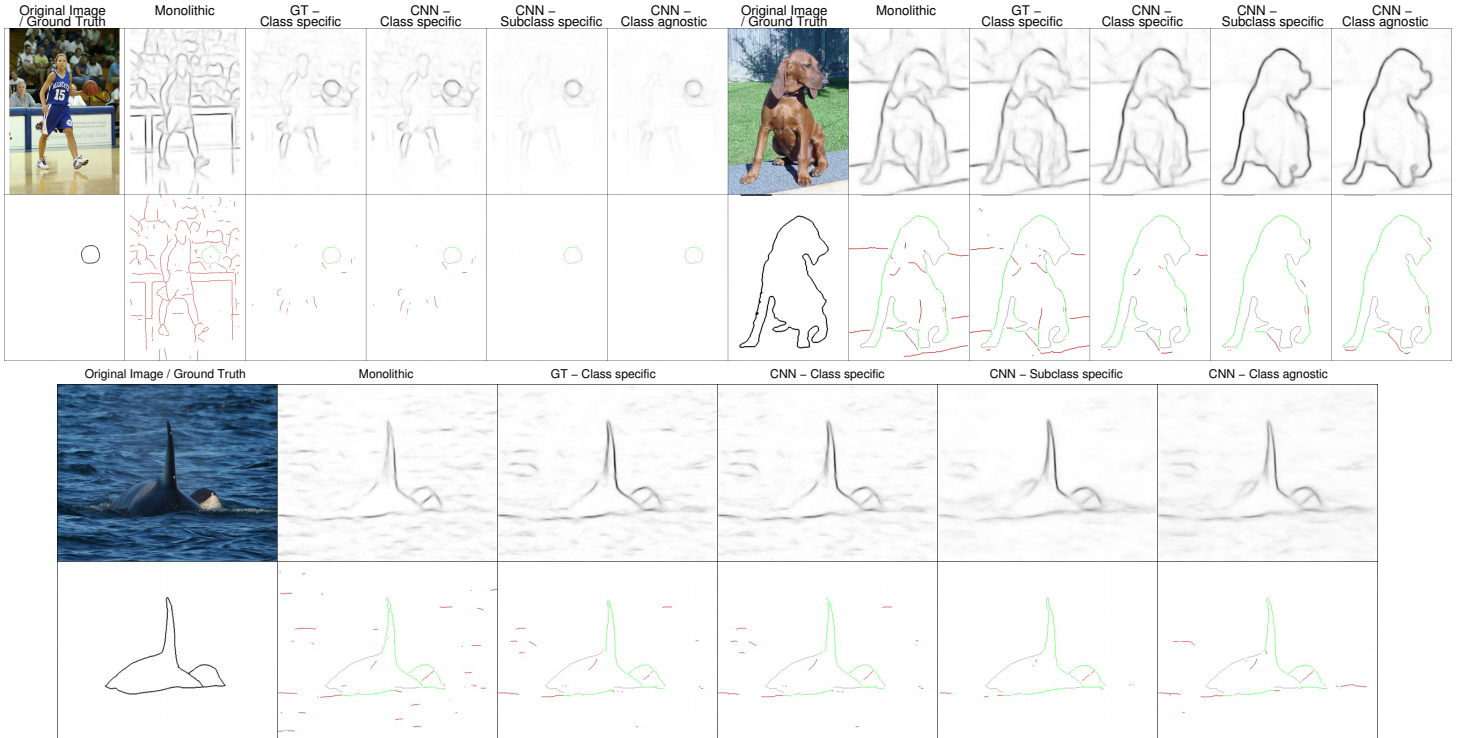


Figure 5. *Qualitative comparison for monolithic versus situational object boundary detection. The upper row shows object boundary predictions. The lower row shows the ground truth boundaries and evaluation at 50% recall, with true positives in green, false positives in red, and undetected boundaries in grey. Monolithic boundary detection fires on many false object boundaries caused by the background and internal boundaries, while situational object boundary detection focuses much better on the boundaries of the object of interest.*

work. As the table shows, there is almost no difference between *GT - class specific* and *CNN - class specific*. Hence within our framework global image classification achieves what can be maximally expected from it.

CNN features inside the Random Forest. Theoretically, the Random Forests can learn from any features of different modalities. So it would arguably be simpler to directly provide global image features to the Structured Edge Forests and bypass the intermediate step of classifying images into situations. We tried this with CNN features, which are stronger and have a lower dimensionality than Fisher vectors. We name this setting *monolithic - CNN enhanced*. Table 2 shows that this does not work better than the baseline monolithic detector.

4.2. Microsoft COCO

Dataset. Microsoft COCO [24] provides accurate segmentations for its 80 object classes such as *person*, *banana*, *bus*, *cat*, and others. We use v0.9 consisting of 82,783 training and 40,504 validation images. Images contain on average 7.7 different object classes. Since evaluation of boundary predictions is relatively slow by necessity [29], we limit evaluation to the first 5,000 images of the validation set (which comes already randomized).

Number of situations. For our subclass specific situations, we choose 10 subclasses per class, leading to a total of 800 situations. We also use 800 class agnostic situations.

Results. In contrast to the previous experiment, here most images contain multiple object classes. Now the first question is: should we train (sub)class specific object boundary detectors on only the object boundaries of the target class or on the boundaries of all object classes present in the image? Results are shown in Table 3. Interestingly, results are slightly better for true single class object boundary detectors in the theoretical setting where we use the Ground Truth to determine the class label (*GT - class specific*). In contrast, when using CNN features results are slightly better when the detectors are trained on all object boundaries in the images. This suggests that mistakes made by object classification can be partially amended by having object boundary predictors specialized to a certain context rather than to a certain object class. For the rest of this paper, we therefore train situational object boundary detectors always on all object boundaries present in the images of a situation.

Figure 6 compares situational object boundary detection with the monolithic baseline. Whereas a monolithic approach yields an AP of 0.368, our situational approaches yield a substantial higher AP at 0.408, 0.424, and 0.434 for respectively class specific, subclass specific, and class agnostic situations. The best AP improvement is almost 0.07 for class agnostic situations.

As before, subclass specific situations outperform class specific situations. But unlike ImageNet, on COCO the class agnostic situations slightly outperform the subclass specific. This is likely because in our ImageNet subset only

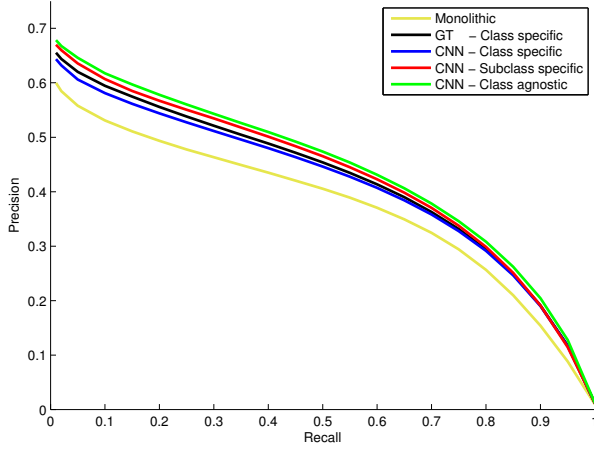


Figure 6. Performance of object boundary detection on the first 5000 images of the COCO validation set.

Detectors trained on class object boundaries only			
	precision at 20% recall	precision at 50% recall	average precision
GT - class specific	0.566	0.460	0.422
CNN - class specific	0.543	0.443	0.407
CNN - subclass specific	0.560	0.459	0.418
Detectors trained on all object boundaries within images			
	precision at 20% recall	precision at 50% recall	average precision
monolithic	0.494	0.406	0.368
GT - class specific	0.556	0.454	0.416
CNN - class specific	0.544	0.446	0.408
CNN - subclass specific	0.567	0.465	0.424
CNN - class agnostic	0.578	0.474	0.434

Table 3. Results on Microsoft COCO. Situational object boundary detection significantly outperforms a monolithic strategy.

a single class is annotated, whereas COCO images often contain multiple classes. The fact that class agnostic situations are superior suggests that the whole context of the image is more important for determining which object boundaries to expect than the specific object classes depicted.

Figure 7 shows qualitative results. In contrast to a monolithic approach, our situational object boundary detector correctly ignores grass/gravel transitions in baseball, contours of buildings (which are not objects of interest) in streets, and interior boundaries of the train.

We conclude that by identifying a situation, we can avoid many false positive object boundary predictions made by a monolithic detector. This leads to significant improvements: whereas a monolithic approach yields 0.368 AP, class agnostic situations yield 0.434 AP, a relative improvement of 18%.

4.3. Pascal VOC 2012 segmentation

Dataset. We use the 1,464 training and 1,449 validation images of Pascal VOC 2012 segmentation, annotated with contours for 20 object classes for all instances in all images.

Number of situations. Since the dataset is a lot smaller than Microsoft COCO, we choose to have 5 subclasses per class to still have sufficient training data per situation, lead-

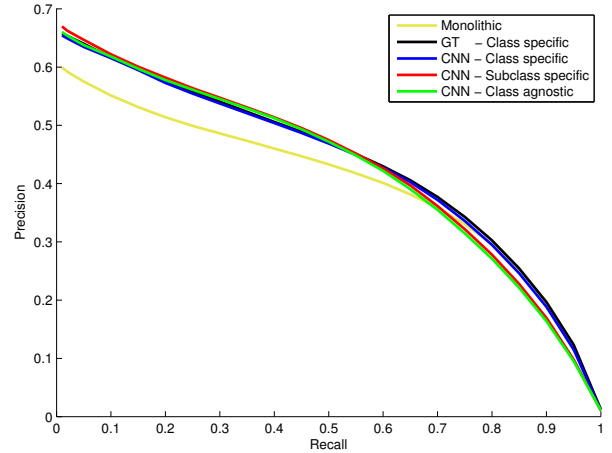


Figure 8. Performance of object boundary detection on the Pascal VOC 2012 segmentation database.

	precision at 20% recall	precision at 50% recall	average precision
monolithic	0.514	0.433	0.396
GT - class specific	0.576	0.470	0.430
CNN - class specific	0.573	0.469	0.426
CNN - subclass specific	0.582	0.475	0.426
CNN - class agnostic	0.578	0.472	0.422

Table 4. Results on validation of Pascal VOC 2012 segmentation.

ing to 100 subclass specific situations. For fair comparison, we also cluster 100 class agnostic situations.

Results. Results are presented in Figure 8 and Table 4. Again, with 0.426 AP the situational object boundary detection significantly outperforms the monolithic performance of 0.396 AP. This is a relative 8% improvement.

On this dataset, class specific situations have about the same performance as subclass specific and class agnostic. This is different than on ImageNet and COCO, most likely because the training set is smaller. Hence fine-grained situations yield fewer benefits since both training appearance based classifiers and training object boundary detectors is more difficult with less data.

4.4. Semantic Boundaries Dataset (SBD)

In some applications one may want to ‘semantic contour detection’ [17], i.e. generating class-specific object boundary maps. Our class-specific boundary detectors can produce such maps $D_c(I)$, specific to class c , using (1) but with the summation running only over class $j = c$:

$$D_c(I) = P(S_c|I) \cdot D_c(I) \quad (3)$$

where $P(S_c|I)$ is the probability that class c occurs in image I according to CNN-based classification. $D_c(I)$ is the output of the class-specific boundary predictor for class c .

We use the Semantic Boundaries Dataset of [17], which consists of 11,318 images from the Pascal VOC 2011 trainval dataset, divided in 8498 training and 2820 test images. All instances of its 20 object classes were annotated with accurate figure/ground masks by crowdsourcing. We use the official evaluation software provided by [17].

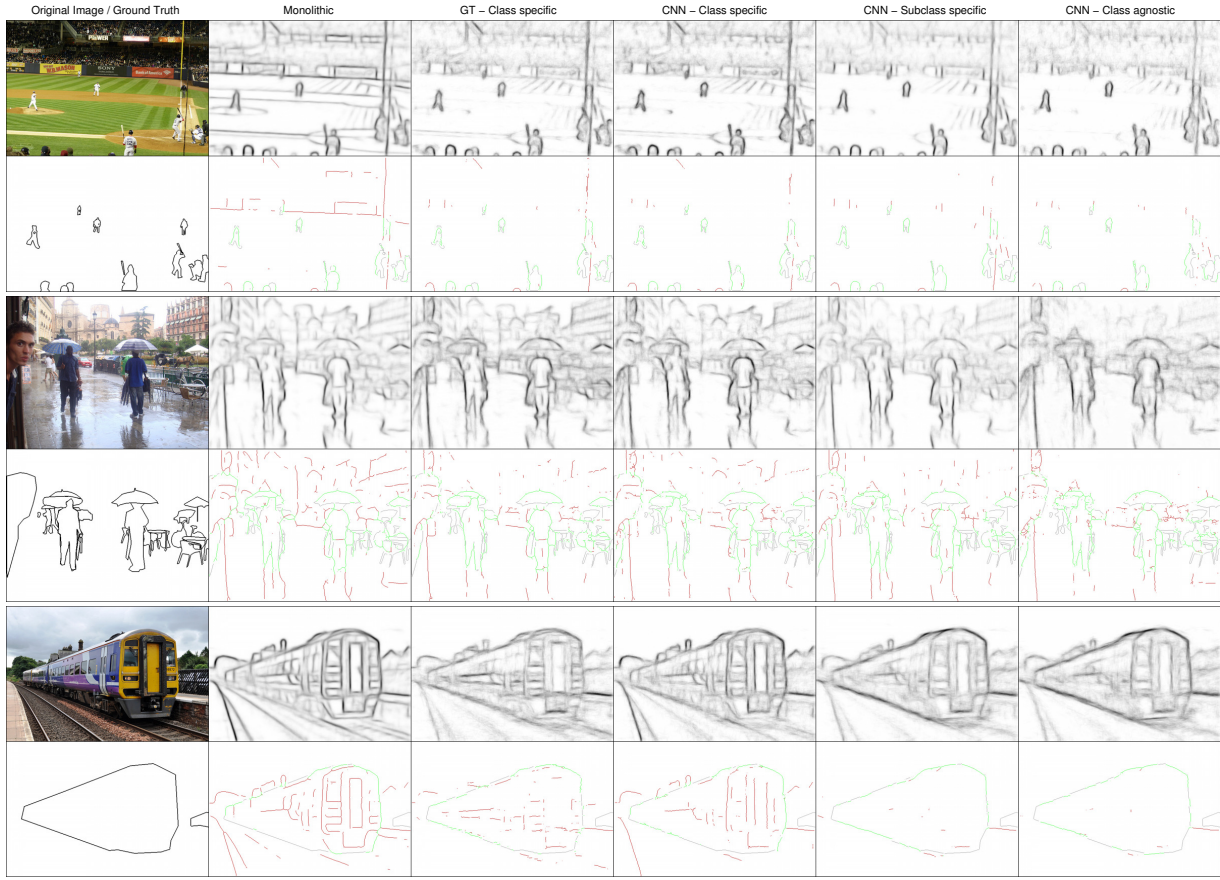


Figure 7. Examples from COCO. Odd rows: input image and boundary predictions. Even rows: ground truth boundaries and precision at a recall of 50%. True positives are green, false positives red, and undetected boundaries grey. While the monolithic detector often incorrectly fires on the background and internal boundaries, our situational object boundary detectors focus better on true object boundaries.

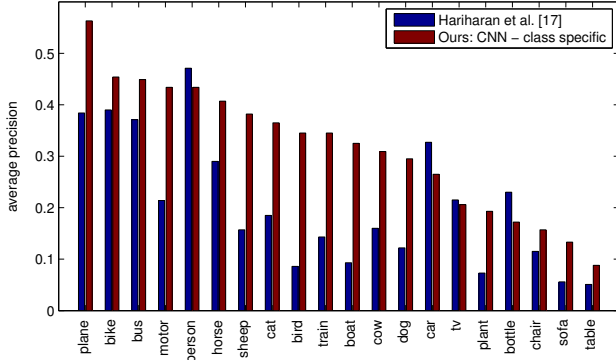


Figure 9. Semantic Contour Detection on BSD. [17] versus our CNN-based class specific situational object boundary detector.

As figure 9 shows our method considerably outperforms [17] on most classes. While [17] report a mean AP of 0.207, we obtain 0.316 mAP.

4.5. Computational Requirements

Runtime on a test image is essentially constant for any reasonable number of situations: the most expensive component is the boundary detector [10] which takes 73 ms/image on an Intel Core i5-3470. At test time we always apply $n = 5$ detectors (Equation (2)). Extracting

CNN features takes about 2 ms/image on a modern GPU. Linear classification on 4096 dimensions takes less than 2 ms/image for 1000 situations. Hence our situational object boundary prediction takes around 0.37 s/image, which is still very fast for an object boundary detector (see e.g. [10]).

5. Conclusion

The appearance of true object boundaries varies from situation to situation. Hence a monolithic object boundary prediction approach which predicts object boundaries regardless of the image content is necessarily suboptimal. Therefore this paper introduces situational object boundary detection. First the situation is determined based on global image appearance. Afterwards only those boundary detectors are applied which are specialized for this situation. Since we build on [10], our situational object boundary prediction is fast and takes only 0.37 ms/image. More importantly, results on object boundary detection show consistent improvements on three large datasets: on Pascal VOC 2012 segmentation [13], the automatically segmented ImageNet [16, 35], and Microsoft COCO [24], we obtained relative improvements of respectively 8%, 14% and 18% AP. Furthermore, on semantic contour detection our approach substantially outperforms [17] on their SBD dataset.

Acknowledgements. This work was supported by the ERC Starting Grant VisCul.

References

- [1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *TPAMI*, 2014. 4
- [2] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *TPAMI*, 2011. 1, 2
- [3] X. Boix, J. Gonfaus, J. van de Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony potentials: Fusing global and local scale for semantic image segmentation. *IJCV*, 2012. 2
- [4] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 2
- [5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 2
- [6] J. Canny. A computational approach to edge detection. *TPAMI*, 1986. 2
- [7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional networks. In *BMVC*, 2014. 5
- [8] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004. 3
- [9] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 2
- [10] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 1, 2, 4, 5, 8
- [11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013. 4, 5
- [12] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. 1973. 2
- [13] M. Everingham, S. Eslami, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge - a retrospective. *IJCV*, 2014. 1, 2, 4, 8
- [14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010. 2
- [15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 4
- [16] M. Guillaumin, D. Küttel, and V. Ferrari. ImageNet auto-annotation with segmentation propagation. *IJCV*, 2014. 2, 4, 8
- [17] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 1, 2, 4, 7, 8
- [18] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009. 2
- [19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 4
- [20] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *ICCV*, 2005. 3
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 4
- [22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006. 3
- [23] J. Lim, C. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 1, 2
- [24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 4, 6, 8
- [25] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR*, 2009. 2
- [26] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004. 3
- [27] J. Mairal, M. Leordeanu, and F. Bach. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008. 2
- [28] D. Marr and E. Hildreth. Theory of edge detection. In *Proceedings of the Royal Society of London*, 1980. 2
- [29] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*, 2004. 2, 4, 6
- [30] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *ECCV*, 2010. 3
- [31] M. Prasad, A. Zisserman, and A. Fitzgibbon. Learning class-specific edges for object detection and segmentation. *Computer Vision, Graphics and Image Processing*, 2006. 2
- [32] J. Prewitt. *Picture Processing and Psychopictorics*, chapter Object Enhancement and Extraction. 1970. 2
- [33] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop at CVPR*, 2014. 4, 5
- [34] L. Roberts. Machine perception of three dimensional solids. In *Optical and Electro-Optical Information Processing*, 1965. 2
- [35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1, 2, 4, 8
- [36] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *TPAMI*, 2000. 2
- [37] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003. 3
- [38] A. Torralba. Contextual priming for object detection. *IJCV*, 2003. 2
- [39] J. R. R. Uijlings, A. W. M. Smeulders, and R. J. H. Scha. Real-time Visual Concept Classification. *IEEE Transactions on Multimedia*, 12, 2010. 3

- [40] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM MM*, 2010. 4